# Algèbre linéaire

Les fonctions d'algèbre linéaire de Maple sont disponibles dans le package `linalg`. Pour charger ces fonctions :

```
> with(linalg):
```

## 1 Vecteurs

Maple ne connaît qu'une seule famille d'espaces vectoriels : les espaces $\mathbb{C}^n$. Ce n'est pas gênant en pratique puisque tout espace vectoriel de dimension finie sur $\mathbb{R}$ ou $\mathbb{C}$ peut être identifié à un sous-espace vectoriel de $\mathbb{C}^n$ pour un certain $n \in \mathbb{N}^*$, modulo le choix d'une base. Par exemple, dans $\mathbb{R}_3[X]$, le sous-espace vectoriel engendré par $X+2$ et $X^3-1$ peut être vu comme le sous-espace vectoriel de $\mathbb{R}^4$ engendré par $(2,1,0,0)$ et $(-1,0,0,3)$ ; pour cette identification, on a travaillé ici avec la base canonique $(1, X, X^2, X^3)$ de $\mathbb{R}_3[X]$.

On peut transformer une liste en un vecteur grâce à la fonction `vector` — qui produit des objets de type `vector` :

```
> v:=vector([1,3,5,7]);
```

$$v := [1, 3, 5, 7]$$

Attention ! Remarquez bien que Maple distingue les listes des vecteurs — alors qu'il les affiche de la même manière.

```
> type([1,2,3],list);
  type([1,2,3],vector);
  type(v,vector);
```

$$true$$
$$false$$
$$true$$

On peut créer un vecteur formel pour faire des calculs formels de la manière suivante :

```
> x:=vector(4);
  evalm(x);
```

$$x := \mathsf{array}(1..4, [\,])$$
$$[x_1, x_2, x_3, x_4]$$

Notez bien que la fonction `evalm` a pour but d'évaluer les quantités vectorielles et de les afficher explicitement, tout comme `evalf` sert à convertir et afficher les nombres sous forme flottante. Cette fonction nous permet en particulier de calculer des combinaisons linéaires.

```
> y:=vector(4);
  evalm(2*x+3*y);
```

$$y := \mathsf{array}(1..4, [\,])$$
$$[2x_1 + 3y_1, 2x_2 + 3y_2, 2x_3 + 3y_3, 2x_4 + 3y_4]$$

## 2 Sous-espaces vectoriels

Maple ne connaît pas à proprement parler la notion d'espace vectoriel ni celle de sous-espace vectoriel. Pour certains problèmes, on peut cependant s'en servir comme s'il connaissait ces notions.

Par exemple, comment trouver une base de $E = \mathbf{Vect}\Big((1,1,1),(1,2,3),(2,3,4)\Big)$ dans $\mathbb{R}^3$ ? — Ici bien sûr, le résultat saute aux yeux. — On utilise la fonction `basis`.

```
> E:=[vector([1,1,1]),vector([1,2,3]),vector([2,3,4])];
              # On identifie sous-espace vectoriel et famille génératrice
  basis(E);
```

$$E := [[1,1,1],[1,2,3],[2,3,4]]$$
$$[[1,1,1],[1,2,3]]$$

Notez bien que $E$ est une famille de vecteurs ci-dessus, pas un sous-espace vectoriel. Pourtant nous considérerons quant à nous que $E$ est le sous-espace vectoriel $\mathbf{Vect}\Big((1,1,1),(1,2,3),(2,3,4)\Big)$. Par ce tour de passe-passe intuitif on peut dire que Maple connaît la notion de sous-espace vectoriel.

On peut aussi calculer une base d'une intersection et d'une somme de sous-espaces vectoriels au moyen des fonctions `intbasis` et `sumbasis`.

```
> F:=[vector([0,1,1]),vector([0,0,1])];
  intbasis(E,F);
  sumbasis(E,F);
```

$$\{[0,1,2]\}$$
$$[[1,1,1],[1,2,3],[0,1,1]]$$

# 3 Matrices

Pour transformer une liste de listes en matrice de type `matrix`, on utilise la fonction `matrix`.

```
> matrix(2,3,[1,2,3,4,5,6]);
  matrix([[1,2,3],[4,5,6]]);
```

$$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$$
$$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$$

On peut créer une matrice formelle pour faire des calculs formels de la manière suivante :

```
> M:=matrix(2,4);
  evalm(M);
```

$$M := mboxarray(1..2, 1..4, [\,])$$
$$\begin{bmatrix}M_{1,1}&M_{1,2}&M_{1,3}&M_{1,4}\\M_{2,1}&M_{2,2}&M_{2,3}&M_{2,4}\end{bmatrix}$$

Pour faire des combinaisons linéaires de matrices et des produits vectoriels, on a ici aussi recours à la fonction `evalm`. Notez bien que le produit matriciel se note « `&*` ». Autres fonctions indispensables, aux noms évocateurs : `transpose`, `inverse`, `det` et `trace`.

```
> A:=matrix(2,2,[1,7,5,8]);
  B:=matrix(2,2,[9,5,1,2]);
  evalm(2*A-B),evalm(A&*B),transpose(A),inverse(B),det(A),trace(B);
```

$$A := \begin{bmatrix}1&7\\5&8\end{bmatrix}$$
$$B := \begin{bmatrix}9&5\\1&2\end{bmatrix}$$
$$\begin{bmatrix}-7&9\\9&14\end{bmatrix},\ \begin{bmatrix}16&19\\53&41\end{bmatrix},\ \begin{bmatrix}1&5\\7&8\end{bmatrix},\ \begin{bmatrix}\frac{2}{13}&-\frac{5}{13}\\-\frac{1}{13}&\frac{9}{13}\end{bmatrix},\ -27,\ 11$$

Les matrices et les vecteurs pour Maple sont en fait issus d'un même type d'objets qu'on appelle des tableaux — type `array`. On peut construire aussi des matrices avec la fonction `array`, et surtout donner à ces matrices des formes particulières : `symmetric`, `antisymmetric`, `diagonal`, `sparse` (matrice nulle) ou `identity`.

```
> U:=array(1..3,1..3,symmetric);
  U[1,1]:=1;
  U[1,2]:=2;
  evalm(U);
```

$$U := \mathsf{array}(symmetric, 1..3, 1..3, [\,])$$
$$U_{1,1} := 1$$
$$U_{1,2} := 2$$
$$\begin{bmatrix}1&2&U_{1,3}\\2&U_{2,2}&U_{2,3}\\U_{1,3}&U_{2,3}&U_{3,3}\end{bmatrix}$$

```
> V:=array(1..3,1..3,sparse);
  V[2,1]:=3;
  evalm(V);
```

$$V := \mathsf{array}(sparse, 1..3, 1..3, [\,])$$
$$V_{2,1} := 3$$
$$\begin{bmatrix} 0 & 0 & 0 \\ 3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Pour finir, n'oublions pas que toute matrice peut être vue comme une application linéaire. Le rang, le noyau et l'image d'une matrice sont accessibles respectivement à partir des fonctions `rank`, `kernel` et `colspace` — le noyau et l'image sont donnés sous la forme de bases.

```
> W:=matrix(3,3,[1,4,3,7,3,0,8,7,3]);
  rank(W);
  kernel(W);
  colspace(W);
```

$$W := \begin{bmatrix} 1 & 4 & 3 \\ 7 & 3 & 0 \\ 8 & 7 & 3 \end{bmatrix}$$
$$2$$
$$\{[9, -21, 25]\}$$
$$\{[1, 0, 1], [0, 1, 1]\}$$